UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/686,746 | 10/16/2003 | Takeshi Tanaka | 82478-1300 | 8702 |

21611 7590 01/08/2008

SNELL & WILMER LLP (OC)
600 ANTON BOULEVARD
SUITE 1400
COSTA MESA, CA 92626

| EXAMINER | |
|---|---|
| DOLLINGER, TONIA LYNN MEONSKE | |
| **ART UNIT** | **PAPER NUMBER** |
| 2181 | |
| **MAIL DATE** | **DELIVERY MODE** |
| 01/08/2008 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/686,746 | TANAKA ET AL. | |
| | Examiner | Art Unit | |
| | Tonia LM Dollinger | 2181 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *15 October 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1,3-10,14 and 16-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-10,14 and 16-21* is/are rejected.

7)☒ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____is/ are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

### *Claim Rejections - 35 USC § 102*

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

2. Claims 1, 3-10, 14, and 16-21 are rejected under 35 U.S.C. 102(b) as being anticipated by Eickemeyer et al (U.S. Patent # 5,355,460), herein referred to as Eickemeyer.

3. As per **claim 1**, Eickemeyer discloses a parallel execution processor comprising:

a plurality of processing elements (See figure 1: Function Units 13-15);

an obtaining unit operable to obtain an instruction sequence including one or more instructions (See figure 1 and column 6, lines 11-17: The compound instruction cache 12 hold sequences of instructions);

a decoding unit operable to decode the obtained instruction sequence into the one of more instructions (See figure 8 and column 13, lines 53-64: Decoders 40, 41, and 45 are capable of sorting out the various op code to group instructions together);

a group forming unit (See figure 1: Instruction Compounding Unit 11) operable to form the processing elements into as many groups as the number of instructions included in the instruction sequence (See column 9, lines 44-49: The ideal case is when there is as many processing elements as instruction sequences); and

an execution controlling unit operable to assign the one or more instructions decoded by the decoding unit to the groups of the processing elements (See figure 8 and column 13, lines 53-64: Decoders 40, 41, and 45 are capable of sorting out the various op code to group instructions together), so that each group of processing elements receives a different one of the one or more instructions, and control the processing elements so that (i) the instructions received by the groups are executed in parallel (Figure 1, See column 5, lines 30-35: The instructions are grouped in such a manner such that the can be run in parallel on the different functional units.), and (ii) in each group, all processing elements in the group each execute the same instruction received by the group (See column 5, lines 60-64: The functional units run in parallel. All of the processing elements, or components, that comprise a functional unit all work together to execute the same instruction received by the entire functional unit, see Figure 1, elements 13-15).

4. **Claim 2** has been cancelled.

5. As per **claim 3**, Eickemeyer discloses the parallel execution processor of claim 1, wherein

when the number of instructions included in the instruction sequence is one, the group forming unit forms all of the processing elements into one group (See column 9, lines 44-49: This is the case when N = 1), and

when the number of instructions included in the instruction sequence is two, the group forming unit forms all of the processing elements into two groups so that the two groups contain an equal number of processing elements (See column 9, lines 44-49: This is the case when N = 2).

6. As per **claim 4**, Eickemeyer discloses the parallel execution processor of claim 3, further comprising

a plurality of register files (See figure 6: Tagged instruction register 27) each of which corresponds to a different one of the processing elements (See column 11, lines 32-38: Registers are tagged for each instruction group), wherein

the instruction sequence includes a first instruction and a second instruction (See column 9, lines 44-49: This is the case when N = 2),

the register files are arranged in the register so that first-group register files and second-group register files alternate (See figure 6 and 9 and column 9, lines 44-49: This would be the case where N = 2 and the choice would be either tag0 or tag1), (i) the first-group register files each storing therein a piece of data to be processed when the first instruction is executed (See figure 6 and 9 and column 9, lines 44-49, and column 11, lines 32-38: This would be the case where N = 2 and the choice would be either tag0 or tag1. Instructions associated with tag0 will be stored in corresponding registers) and (ii) the second-group register files each storing therein a piece of data to be processed when the second instruction is executed (See figure 6 and 9, column 9, lines 44-49:

This would be the case where N = 2 and the choice would be either tag0 or tag1. Instructions associated with tag1 will be stored in corresponding registers),

when the number instructions included in the instruction sequence is two, the group forming unit forms the processing elements corresponding to the first-group register files into one of the two groups, and the processing elements corresponding to the second-group register files into the other group (See column 9, lines 44-49: This is the case when N = 2), and

each of the processing elements obtains the piece of data to be processed from the corresponding register file (See figure 1 and column 6, lines 11-17: Only the corresponding functional unit will process the group data in the registers).

7. As per **claim 5**, Eickemeyer discloses the parallel execution processor of claim 4, wherein

the register files are formed into a plurality of pairs (See column 10, lines 17-19: Pairwise compounding keeps ensures pairs of register files), keeping an order in which the register files are arranged (See column 11, lines 32-38: Tags are used to ensure that register files are kept in order),

each of the instructions includes a piece of selection information indicating which piece of data each processing element should obtain (See figure 6: Each register file has instructions), selecting out of (a) the piece of data stored in the corresponding register file and (b) the piece of data stored in a register file with which the corresponding register file is paired (See column 12, lines 22-44: Data can be grabbed

from another group if needed but instruction will be run if no dependencies are outstanding), and

each of the processing elements obtains the piece of data to be processed from the register file indicated in each piece of selection information (See figure 1 and column 6, lines 11-17: Only the corresponding functional unit will process the group data in the registers).

8. As per **claim 6**, Eickemeyer discloses the parallel execution processor of claim 3, wherein

when the number of instructions included in the instruction sequence is two, the execution controlling unit includes:

a storing unit (See column 13, lines 16-20: The compound analyzer 22 stores the combination options) that stores therein a plurality of combination options based on which of the processing elements should belong to each of the two groups (See column 13, lines 23-31: Examples of combination rules), the combination options being prepared for each of a plurality of grouping procedures;

a grouping information obtaining unit (See figure 1: Instruction Compounding Unit 11) operable to obtain a piece of grouping information indicating which one of the grouping procedures should be used (See column 11, lines 32-38: Tags are used to identify groups); and

a selecting unit operable to select one of the combination options according to the obtained piece of grouping information (See figure 1 and column 6, lines 11-17: Only the corresponding functional unit will process the group data in the registers).

9. As per **claim 7**, Eickemeyer discloses the parallel execution processor of claim 3, wherein

when the number of instructions included in the instruction sequence is two, the execution controlling unit includes:

a grouping information obtaining unit (See figure 1: Instruction Compounding Unit 11) operable to obtain a piece of grouping information indicating to which one of the two groups, each of the processing elements should belong (See column 11, lines 32-38: Tags are used to identify groups); and

a grouping unit operable to form the processing elements into the two groups according to the obtained piece of grouping information (See column 9, lines 44-49: This is the case when N = 2).

10. As per **claim 8**, Eickemeyer discloses the parallel execution processor of claim 1, further comprising

a fetching unit (See figure 10: Fetch/Issue Control Unit) operable to fetch a piece of data which is of a predetermined length and has a format field and a data field (See column 15, line 68- column 16, line 11: The fetch unit utilizes op code, which has a predetermined length, format and data field, to fetch data), wherein

each of the instructions includes an OP code and an operand (See column 15, line 68- column 16, line 11: Op code is used),

a positioning pattern is written in the format field (See figure 7 and column 11, lines 32-38: Tags are used to identify instructions, which are held in the instruction register 21), the positioning pattern being for positioning OP codes and operands in the data field (See column 11, lines 32-38: Tags are used to identify groups and used to place instructions into the corresponding groups),

in the piece of data, one or more OP codes and one or more operands are arranged in the data field in an order defined by the positioning pattern written in the format field (See figure 7: The instruction register 21 can hold more than one instruction and the system is able to identify the start on one by its tag),

the obtaining unit obtains, as the instruction sequence, the piece of data of the predetermined length fetched by the fetching unit (See figure 1 and column 6, lines 11-17: The compound instruction cache 12 fetches and holds sequences of instructions),

the decoding unit extracts, from the piece of data, the one or more OP codes and the one or more operands, according to the positioning pattern so as to decode the OP codes and the operands of the instructions (See figure 8 and column 13, lines 53-64: Decoders 40, 41, and 45 are capable of sorting out the various op code to group instructions together), and

the execution controlling unit assigns, in the defined order, the decoded instructions to the groups (See column 5, lines 30-35: The instructions are grouped by encoding in their tags).

11. As per **claim 9**, Eickemeyer discloses the parallel execution processor of claim 1, further comprising:

a fetching unit (See figure 10: Fetch/Issue Control Unit) operable to fetch a piece of data which is of a predetermined length (See column 15, line 68- column 16, line 11: The fetch unit utilizes op code, which has a predetermined length); and

a storing unit operable to store therein a predetermined positioning pattern for OP codes and operands (See figure 1: Compound Instruction cache 12), wherein

each of the instructions includes an OP code and an operand (See column 15, line 68- column 16, line 11: Op code is used),

one or more OP codes and one or more operands are arranged in the piece of data in an order defined by the predetermined positioning pattern (See column 11, lines 32-38: Tags are used to identify groups and used to place instructions into the corresponding groups),

the obtaining unit obtains, as the instruction sequence, the piece of data of the predetermined length fetched by the fetching unit (See figure 1 and column 6, lines 11-17: The compound instruction cache 12 fetches and holds sequences of instructions),

the decoding unit extracts, from the piece of data, the one or more OP codes and the one or more operands, according to the positioning pattern stored in the storing unit so as to decode the OP codes and the operands of the instructions (See figure 8 and column 13, lines 53-64: Decoders 40, 41, and 45 are capable of sorting out the various op code to group instructions together), and

the execution controlling unit assigns, in the defined order, the decoded instructions to the groups (See column 5, lines 30-35: The instructions are grouped by encoding in their tags).

12. As per **claim 10**, Eickemeyer discloses the parallel execution processor of claim 1, wherein

when the instruction sequence obtained by the obtaining unit includes two or more instructions and one of the instructions instructs that processing elements included in some of the groups should halt operation, the execution controlling unit controls the processing elements included in those groups so that those processing elements halt operation (See column 12, lines 22-44: A halt is necessary when a dependency is found, which would halt the group).

13. **Claims 11-13** have been cancelled.

14. **Claim 14** is rejected for reasons similar to that of claim 1. Claim 14 is the method of the parallel execution processor of claim 1.

15. **Claim 15** has been cancelled.

16. As per **claim 16**, Eickemeyer discloses a parallel execution processor comprising:

a plurality of processing elements (See figure 1: Function Units 13-15);

an obtaining unit operable to obtain an instruction sequence including one or more instructions (See figure 1 and column 6, lines 11-17: The compound instruction cache 12 hold sequences of instructions), wherein the number of processing elements is greater than the number of instructions (See column 19, lines 9-16: The number of instructions may vary and whereas Eickemeyer described the ideal situations in detail, other cases may occur);

a decoding unit operable to decode the obtained instruction sequence into the one of more instructions (See figure 8 and column 13, lines 53-64: Decoders 40, 41, and 45 are capable of sorting out the various op code to group instructions together);

a group forming unit (See figure 1: Instruction Compounding Unit 11) operable to form the processing elements into as many groups as the number of instructions included in the instruction sequence (See column 9, lines 44-49: The ideal case is when there is as many processing elements as instruction sequences); and

an execution controlling unit for assigning the one or more instructions decoded by the decoding unit to the groups of the processing elements (See figure 8 and column 13, lines 53-64: Decoders 40, 41, and 45 are capable of sorting out the various op code to group instructions together), so that each group of processing elements receives a different one of the one or more instructions, and control the processing elements so that (i) the instructions received by the groups are executed in parallel (See column 5, lines 30-35: The instructions are grouped in such a manner such that the can be run in parallel), and (ii) in each group, all processing elements in the group each execute in

parallel the same instruction received by the group (See column 5, lines 60-64: The functional units run in parallel. All of the processing elements, or components, that comprise a functional unit all work together to execute the same instruction received by the entire functional unit, see Figure 1, elements 13-15).

17. As per **claim 17**, Eickemeyer discloses the parallel execution processor system of Claim 16 further comprising at least two processing elements and wherein the obtaining unit obtains an instruction sequence including only one instruction (See column 19, lines 9-16: The number of instructions may vary and whereas Eickemeyer described the ideal situations in detail, other cases may occur such as when there is two processing elements and only one instruction).

18. As per **claim 18**, Eickemeyer discloses further including a source of a plurality of instructions (See abstract: The instructions are fetched from memory).

19. **Claims 19-21** are rejected for reasons similar to claims 16 and 17. These claims differ in the amount of processing elements and maximum instructions. Eickemeyer does not teach a specific number and allows for the variations claimed.

***Response to Arguments***

20. Applicant's arguments filed October 15, 2007 have been fully considered but they are not persuasive.

21. On page 10, Applicant argues in essence:

*"Eickemeyer does not teach or suggest "a group forming unit operable to form the processing elements into as many groups as the number of instructions included in the instruction sequence." Eickerneyer does not form the processing elements into as many groups as the number of instructions in the instruction sequence. Eickerneyer does not form processing elements into groups at all. Eickemeyer merely determines whether an instruction can be executed in parallel or not. If the instruction can be executed in parallel, the instruction is tagged with a "0" bit and the instruction is executed after the instruction tagged with a "1" bit."*

However, the processing elements are formed into as many active execution groups as there are instructions that can be executed in parallel. For example if there are two instructions, then two groups of processing elements, or functional units, are required to execute the instructions and thus two groups of processing elements are formed. The active execution groups are formed when the instructions are issued to the various functional units, see Figure 1. The ideal case is when N groups are formed. Therefore this argument is moot.

22. On page 11, Applicant argues in essence:

*"However, in the present invention, to realize N-parallel instructions, the present invention does not need to fetch N-instructions.*

*...*

*Thus, the present invention can realize N-parallel execution by fetching two instructions. Therefore, the present invention can realize N-parallel execution without fetching N instructions."*

In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant

relies (i.e., realizing N-parallel execution without fetching N instructions) are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

23. On page 12, Applicant argues in essence:

*"However, Eickemeyer does not teach forming the processing elements in groups and thus the instructions are not assigned to groups. Instead, decoders 40, 41, and 45 determine whether instructions can be processed in parallel. Thus, decoders 40, 41, and 45 determine whether the instructions are in category A or B. Then, they determine based on the rules recited in Column 13, lines. 25 - 29 whether the instructions can be processed in parallel or not. If the instructions can be processed in parallel, the first instruction is tagged with a "1" bit and the second instruction is tagged with a "0" bit. Otherwise, the instructions are tagged with a "1" bit only. If the following instruction is tagged with a "1" bit, then both instructions are executed singularly instead of in parallel. Thus, in Eickemeyer, the instructions are not assigned to groups of processing elements, but only bit tags to determine whether they can be processed in parallel or not."*

However, as described above, the processing elements are formed into as many active execution groups as there are instructions that can be executed in parallel. For example if there are two instructions, then two groups of processing elements, or functional units, are required to execute the instructions and thus two groups of processing elements are formed. The active execution groups are formed when the instructions are issued to the various functional units, see Figure 1. The ideal case is when N groups are formed. Therefore this argument is moot.

24. On page 12, Applicant argues in essence:

*"In Eickemeyer not all of the processors are utilized since there are some instructions which are processed singularly such as when back to back instructions have a "1" bit. There is also no teaching in Eickemeyer that in each group of processing elements, all of the processing elements execute the same instruction received by the group since Eickemeyer does not break the processing elements into groups and therefore each of the processing elements in the non-existent group cannot process the same instruction."*

However, as described above the processing elements are formed into groups. All of the processing elements, or components, that comprise a functional unit all work together to execute the same instruction received by the entire functional unit, see Figure 1, elements 13-15. Therefore this argument is moot.

25. On page 11, Applicant argues in essence:

*"With respect to Claim 3, Eickemeyer fails to recite "when the number of instructions included in the instruction sequence is one, the group forming unit forms all of the processing elements into one group." The Office Action cited to "N = i" for the proposition that the group forming unit forms all of the processing elements in one group. However, "N" represents the number of instructions in the group which are being processed in parallel and furthermore represents the speed of the computer system and not the number of processors used. If N = 1, then it only means that there is one instruction being processed at a time. That is if, there are two processors, one processor would sit idle in Eickemeyer. In the present invention, however, if there is only one instruction, all of the processing elements would process the single instruction and no processing element would sit idle. Thus, the structure in Eickemeyer is different from the structure of the present invention."*

In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., if there is only one instruction, all of the processing elements would process the single instruction and no processing element would sit idle) are not

recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns*, 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993). Therefore this argument is moot.

***Conclusion***

26. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

27. A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

28. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tonia LM Dollinger whose telephone number is (571) 272-4170. The examiner can normally be reached on Monday-Friday with first Friday's off.

29. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Alford Kindred can be reached on (571) 272-4037. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

30. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TLMD

Tonia L. M. Dollinger
Primary Examiner
January 5, 2007